A Comparative Study of Autoregressive Neural Network Hybrids

Tugba Taskaya-Temizel, Matthew C. Casey
University of Surrey
School of Electronics and Physical Sciences
Department of Computing
Guildford, UK
E-mail: t.taskaya, m.casey@surrey.ac.uk

***Abstract*— Many researchers have argued that combining many models for forecasting gives better estimates than single time series models. For example, a hybrid architecture comprising an autoregressive integrated moving average model (ARIMA) and a neural network is a well-known technique that has recently been shown to give better forecasts by taking advantage of each model's capabilities. However, this assumption carries the danger of underestimating the relationship between the model's linear and non-linear components, particularly by assuming that individual forecasting techniques are appropriate, say, for modeling the residuals. In this paper, we show that such combinations do not necessarily outperform individual forecasts. On the contrary, we show that the combined forecast can underperform significantly compared to its constituents' performances. We demonstrate this using nine data sets, autoregressive linear and time-delay neural network models.**

I. Introduction

Research in time series forecasting argues that predictive performance improves in combined models (Bishop, 1994; Clemen, 1989; Hansen & Nelson, 2003; Hibbert, Pedreira, & Souza, 2000; Terui & van Dijk, 2002; Tseng, Yu, & Tzeng, 2002; Weigend, Mangeas, & Srivastava, 1995; Zhang, 2003; Zhang & Qi, 2005). The motivation for combining models comes from the assumption that either one cannot identify the true data generating process (Terui & van Dijk, 2002) or that a single model may not be sufficient to identify all the characteristics of the time series (Zhang, 2003). For example, a time series may exhibit both linear and non-linear patterns during the same time interval. In such cases, neither a linear nor non-linear model is able to model both components simultaneously.

Using a hybrid technique that decomposes a time series into its linear and non-linear form has recently been shown to be successful for single models (Zhang, 2003; Zhang & Qi, 2005). In particular, it has been argued that for seasonal time series, the seasonal component is first required to be removed by a linear model, such as a seasonal autoregressive process, before any further analysis takes place (Tseng et al., 2002; Zhang & Qi, 2005; Nelson, Hill, Remus, & O'Connor, 1999; Virili & Freisleben, 2000). However, this assumption carries the danger of underestimating the relationship between the components as there may not be any additive association between the linear and non-linear elements. In addition, one cannot guarantee that the residuals of the linear component may comprise valid non-linear patterns. Nevertheless, a single component is able to model such seasonal series if the modeling procedure is carried out properly. In this paper, we present a comparison of the performance of these approaches, expanding upon our preliminary work (Taskaya-Temizel & Ahmad, 2005)[1].

In Section II we first discuss the hybrid techniques designed for time series analysis. In Section III we present single models to analyze seasonal time series. Section IV describes the experimental model design, whilst Section V details the experiments and results. Finally, we conclude this work and discuss our future work in Section VI.

II. Model Combination Techniques

There are a range of combination techniques that can be applied to forecasting that attempt to overcome the deficiencies of single models. The difference between these combination techniques can be described using terminology developed for the classification and neural network literature (Sharkey, 2002). Here we focus upon cooperative ensembles and more general cooperative and competitive architectures.

In an ensemble architecture, the aim is to reduce the risk of using an inappropriate model by combining several to reduce the risk of failure. Typically this is done because the underlying process cannot easily be determined (Hibon & Evgeniou, 2005). Ensemble architectures comprise several redundant models designed for the same function, where the *diversity* of the components is thought important (Brown, Wyatt, Harris, & Yao, 2005). An overall forecast is produced by combining the models' outputs, say by an average or majority vote. Ensemble models can be homogeneous, such as using differently configured neural networks (all multilayer perceptrons) (Zhang & Berardi, 2001), or heterogeneous, such as with both linear and non-linear models (Terui & van Dijk, 2002; Wichard & Ogorzalek, 2004). However, these architectures do not always lead to better estimates when compared to single models. For example, it has been shown that combined forecasts do not necessarily dominate for all

[1]An abbreviated version of some portions of this article appeared in Taskaya-Temizel and Ahmad (2005), as part of the IJCNN 2005 conference proceedings, published under the IEEE copyright.

series; sometimes a linear model still produces better results (Terui & van Dijk, 2002).

In a cooperative modular combination, the aim is to fuse models to build a complete picture from a number of partial solutions (Sharkey, 2002). The assumption is that a model may not be sufficient to represent the complete behavior of a time series, for example if the time series exhibits both linear and non-linear features, neither linear models nor non-linear models alone are capable. A good exemplar are models that fuse ARIMA with neural networks. An ARIMA process combines three different processes comprising an autoregressive (AR) function regressed on past values of the process, moving average (MA) function regressed on a purely random process with mean zero and variance σ_t, and an integrated (I) part to make the data series stationary by differencing. In such hybrids, whilst the neural network model deals with non-linearity, the ARIMA model deals with the non-stationary linear component (Tseng et al., 2002; Zhang, 2003; Zhang & Qi, 2005). Such models are generally constructed in a sequential manner, with the ARIMA model first applied to the original time series, and then its residuals modeled using neural networks.

Different hybrids of ARIMA and neural networks have also been constructed. For example, ARIMA parameters have been used as a window to build a neural network architecture (Hansen & Nelson, 2003), whereas neural networks have also been trained with past observations, comprising the original data and ARMA forecasts (Hibbert et al., 2000). However, it is typically assumed that the residuals of a linear component are always going to include valid non-linear patterns that can be modeled using neural networks (Zhang, 2003; Zhang & Qi, 2005). Such assumptions are likely to lead to unwanted degeneration of performance if the opposite situation occurs.

In a competitive architecture the aim is to build appropriate modules to represent different parts of the time series, and to be able to switch control to the most appropriate. For example, a time series may exhibit non-linear behavior generally, but this may change to linearity depending on the input conditions. Early work on threshold autoregressive models (TAR) used two different linear AR processes, each of which change control among themselves according to the input values (Tong, 1990). An alternative is a mixture density model (Bishop, 1994), also known as non-linear gated expert (Weigend et al., 1995), which comprises neural networks integrated with a feedforward gating network. Mixture models have been also extended to comprise Gaussian AR components (Wong & Li, 2000), which work in-situ and are often homogeneous. Whilst each mixture network learns to specialize on different probability density functions of the targets, the gating network learns to switch to the appropriate component based on the input (Jacobs, Jordan, Nowlan, & Hinton, 1991). Such models are thought superior because they can model general conditional densities (Bishop, 1994), whereas conventional neural network approaches approximate the conditional average of the target data by minimizing the sum-of-squares error function. The major drawback of such architectures is that there may be unwanted effects if control is switched to a less well-performing module, thus causing overall performance degeneration.

III. Models for Seasonal Time Series

Many conventional statistical techniques decompose a time series into trends, seasonalities, cycles and irregular fluctuations. Such decomposition facilitates forecasting by providing insights regarding the nature of the time series. The decomposition process comes from the idea that economic theories that are relevant in the long run are different to the theory one wishes to apply in the short run (Harvey, 1997).

Cyclic patterns are oscillations that generally have a fixed period. Seasonality is regarded as a special case of cycles whose periods are calendar fixed. In economic data, there is increasing evidence that business cycles are not symmetric (Chatfield, 2004). Asymmetric cyclic behaviors in the economy can be explained as the rate of change in recession, being different to the rate of change in emerging from recession. Well-known data sets such as the sunspot and Canadian lynx series (Rao & Sabr, 1984) show evidence of asymmetric cycles, with such behavior difficult to model with linear techniques.

If the cyclic patterns are not of direct interest, one can remove them by seasonal differencing conditional on the stochastic variation present in the data. Trend and seasonality removal processes are referred as pre-whitening methods. If the cyclic patterns are of interest, one can apply seasonal models. In the case of cycles that are symmetric, linear AR model variants can be employed, whereas a time series that exhibits multiplicative seasonality can be transformed into additive form using functional transformations such as logarithms (Box & Cox, 1996).

Non-linear models (Kantz & Schreiber, 1999) can also be used to explain, and give forecasts for, data exhibiting regular cyclic behaviors and are an alternative to the use of harmonic components, especially if the behavior is asymmetric (Chatfield, 2004). However, a linear AR model can be applied to a non-linear time series such as to the sunspot data set if the time series is short (Rao & Sabr, 1984). Some empirical results show that linear models dominate in the short run and non-linear models perform well in the long run (Terui & van Dijk, 2002). Moreover, some results show that seasonal series cannot be modeled successfully with neural networks (Zhang & Qi, 2005; Tseng et al., 2002; Nelson et al., 1999). However, no significant attention has been shown to model selection for neural networks and preprocessing in these results.

IV. Model Details

In this paper, our main aim is to investigate whether the performance of hybrid models shows consistent improvement over single models. For this purpose, we compute linear AR, neural network and ARIMA neural network hybrid models, constructed using a range of parameters to determine the best architecture. Our main goal is to evaluate the use of hybrid models and to achieve this, we set out to answer the following questions:

A) How important is preprocessing for neural networks? How does detrending affect the performance?
B) Are neural networks able to model seasonality? If they are, how can we construct optimal architectures?
C) Compared to linear autoregressive models, how successful are neural networks?
D) Are ARIMA neural network hybrids better than single models?

In this section, we present details of the models used to answer these questions.

A. Neural Network Design

Temporal data can be modeled using neural networks in two ways. The first way is to provide recurrent connections from output nodes to the preceding layer (Elman, 1990). The second way is to provide buffers on the output of the nodes (see Haykin (1999) for a detailed survey on neural networks for temporal data modeling). A time-delay neural network (TDNN) is a well-known exemplar for the latter models that has been employed throughout our experiments. In a TDNN, each layer is connected to its preceding layer's buffered output, and is therefore able to relate current input to past values (Waibel, Hanazawa, Hinton, Shikano, & Lang, 1989). A subset of the TDNN architecture is the input delayed neural networks (IDNN), in which the memories are only provided in the input layer (Clouse, Giles, & Horne, 1997). Their simplicity of implementation has made them widely used in time series analysis (Zhang & Berardi, 2001; Zhang, 2003; Zhang & Qi, 2005; Tseng et al., 2002; Weigend et al., 1995).

The activation function for node i at time t of a TDNN is:

$$y_i(t) = f\left(\sum_{j=1}^{M}\sum_{d=1}^{T} w_{ij}(t-d)y_j(t-d)\right) \quad (1)$$

where $y_i(t)$ is the output of node i at time t, $w_{ij}(t)$ is the connection weight between node i and j at time t, T is the number of tapped delays, M is the number of nodes connected to node i from preceding layer, and f is the activation function, typically the logistic sigmoid. In this paper, we consider the case when we have tapped delays in the input layer only.

We consider TDNN configurations of $2i:2j:1$, where $1 \leq i,j \leq 16$ and $i,j \in Z^+$. Each configuration was tested with 30 different random initial conditions to provide an average root mean square error (RMSE) on the test data. Here we focus on RMSE only for model comparison, rather than using other error criteria. Details of the training procedure used can be found in Taskaya-Temizel and Ahmad (2005). Note that the neural networks are trained on normalized data formed using the z-score of the original data.

B. Linear Autoregressive Process Design

A linear AR process has been employed throughout the experiments. A process X_t is said to be an AR process of order p if:

$$X_t = \mu + \alpha_1(X_t - \mu) + ... + \alpha_p(X_{t-p} - \mu) + Z_t \quad (2)$$

where α are the AR parameters, μ is the mean of the series and Z_t is a random process with mean 0 and variance σ_z^2.

As a model selection criterion, we employed Akaike's Information Criterion (AIC), which takes into account the number of parameters fitted. AIC chooses the best fit, as measured by the likelihood function subject to a penalty term (Chatfield, 2004). However, as AIC is biased for small samples, we preferred the bias-corrected version of AICC (Hurvich, Simonoff, & Tsai, 1998):

$$AICC_p = -2ln(\hat{\sigma}_p^2) + 2p + 2p(p+1)/(T-p-1) \quad (3)$$

where T is the sample size, $\hat{\sigma}_p^2 = (T-p-1)^{-1}\sum_{t=p}^{T}\hat{\epsilon}_t^2$ and $\hat{\epsilon}_t$ are the model residuals. The AICC value was calculated for orders between 1 and 20. Then the lowest value was selected among the results.

C. Autoregressive and Neural Network Hybrid Design

A hybrid model comprising a linear and a non-linear component has been employed in the experiments (Zhang, 2003):

$$y_t = L_t + N_t \quad (4)$$

where L_t is the linear AR component and N_t is the non-linear component. First, we model the linear part by fitting an AR function to the data series. Then, the residuals are modeled using neural networks. Let r be the residual of the linear component, then:

$$r_t = y_t - \hat{L}_t \quad (5)$$

where $\hat{L}_t$ is the estimate of the linear AR component. For non-linear patterns, we use neural networks:

$$\hat{r}_t = f(r_{t-1}, r_{t-2}, ..., r_{t-q}) \quad (6)$$

where q is the number of input delays and f is the non-linear function. So the combined forecast will be

$$y_t = \hat{L}_t + \hat{r}_t + \epsilon_t \quad (7)$$

where ϵ_t is the error of the combined model. Since linear AR models cannot model non-linearity, we assume that the residuals of the linear component will contain non-linear patterns, which a non-linear component, such as a neural network, should be able to model. In this way, the hybrid model is exploiting the strength of both components.

V. Experiments and Results

In this section, we describe the experiments and results undertaken to answer the questions set in Section IV. We selected nine monthly time series as used by Zhang and Qi (2005) for the experiments. Monthly series were selected as they exhibit stronger seasonality than that of quarterly time series. None of the series are seasonally adjusted, but do comprise trends (see Table I). All data series end at December 2001. The last 12 values have been reserved for testing, the preceding 12 values for validation, whilst the rest are used for training. This low number of test and validation samples was selected because of the small size of the data sets. It is recognized that this is less than ideal, but is used for comparison with Zhang and Qi (2005), as is one-step-ahead forecasting.

TABLE I

DATA SETS USED IN EXPERIMENTS. THE SECOND COLUMN SHOWS THE START DATE OF THE DATA SERIES. THE LAST COLUMN SHOWS THE TOTAL NUMBER OF DATA POINTS IN THE DATA SETS.

Data Sets	Start Date	Data Points
USBC Retail	01/1992	120
USBC Hardware	01/1992	120
USBC Clothing	01/1992	120
USBC Furniture	01/1992	120
USBC Bookstore	01/1992	120
FR Durable Goods	01/1947	660
FR Fuels	01/1947	576
FR Consumer Goods	01/1970	384
FR Total Production	01/1947	660

A. Experiment 1: How does detrending affect the performance of neural networks?

Although neural networks are said to be universal approximators, they have certain limitations. It has been shown that neural networks are not able to model a time series containing trend, since non-linear transfer functions, such as the logistic sigmoid, constrain the model to the input range values (Cottrell, Girard, Girard, Mangeas, & Muller, 1995). Therefore, it is important to eliminate trend before training, where ideally a stationary time series that has constant mean and variance should be used for modeling. Non-stationarity in the mean attributed to trend can be removed either by differencing (stochastic trends) or polynomial fitting (deterministic trends). However, there is no successful method that determines which detrending method is suitable for a given series (Zhang & Qi, 2005). Although the importance of detrending is known for neural networks, this has yet to be fully investigated.The forecasting ability of neural networks can be helpful in understanding whether differencing or trend fitting can be more appropriate in order to make the time series stationary in the mean.

TABLE II

TDNN MEAN AND STANDARD DEVIATION RMSE FOR TESTING DATA SETS PREPROCESSED WITH DIFFERENCING AND TREND FITTING

Data Sets	Differencing	Trend Fitting
USBC Retail	1446.77±457.84	2177.04±542.28
USBC Hardware	73.26± 20.99	111.06± 31.30
USBC Clothing	1148.58±427.17	848.59±207.98
USBC Furniture	279.91± 44.20	285.97± 24.09
USBC Bookstore	224.86± 36.97	296.08± 38.18
FR Durable Goods	4.31± 0.53	7.83± 1.27
FR Fuels	2.28± 0.30	2.51± 0.54
FR Consumer Goods	1.80± 0.22	2.58± 0.52
FR Total Production	1.95± 0.19	3.42± 1.42

In the literature, both detrending techniques have been applied regardless of observing their performances on testing data sets. For example, whilst Virili and Freisleben (2000) adopted differencing, Zhang and Qi (2005) employed trend fitting. Table II shows the TDNN testing data set RMSE when trained using data preprocessed with differencing or first order polynomial trend fitting, as per Zhang and Qi (2005). The mean result is shown based on training 256 different TDNN architectures for 30 trials, each starting with different random initial conditions. Eight out of nine data sets preprocessed with differencing performed significantly better than with trend fitting. However, we can conclude that one should consider both detrending techniques for modeling with neural networks and choose the best-performing from the results because this will typically depend upon the data set. For the nine data sets, differencing appears to give better results, and hence we use this in the subsequent experiments.

For neural networks, preprocessing helps to make the data have a constant mean and variance. One should expect good forecasts if the data set has been properly adjusted according to the nature of the series before training. The experiments indicate that the trends in the data sets cannot be adequately captured by straight lines, which means a deterministic trend is too restrictive, which is also inline with the Harvey's (1997) result. However, we note that if the time series evolves in exponential or multiplicative form, the first step should be to apply transformations such as taking the logarithm of the series.

B. Experiment 2: How can we construct optimal neural network architectures for seasonal time series?

In this experiment, we investigated whether neural networks are able to model seasonal time series. For each data set, we selected the TDNN configurations that produced the best mean performance (lowest RMSE) out of 256 (see Table III), and compared these with Zhang and Qi's (2005) TDNN model, who determined the number of the input nodes and delays according to the nature of the autocorrelation in the time series. Note that their results report the best-fit model among 5 trials of 98 architectures only.

TABLE III

COMPARISON OF TDNN ARCHITECTURES: THE SECOND COLUMN SHOWS THE BEST TDNN CONFIGURATION OBTAINED FROM 256 MODELS. THE THIRD COLUMN PRESENTS THE MEAN AND STANDARD DEVIATION OF RMSE RESULTS OF CORRESPONDING MODELS BASED ON 30 TRIALS. THE LAST COLUMN (*) SHOWS THE BEST FIT RESULTS OF ZHANG AND QI (2005)

Data Sets	Model	TDNN	TDNN *
USBC Retail	16: 2 :1	628.70±28.27	1785.77
USBC Hardware	14: 4 :1	35.70± 6.90	105.12
USBC Clothing	14: 2 :1	372.50±50.66	1117.72
USBC Furniture	16: 2 :1	173.10±31.10	226.68
USBC Bookstore	12: 2 :1	91.51±10.41	170.49
FR Durable Goods	12:16:1	2.91± 0.34	5.98
FR Fuels	32: 2 :1	1.64± 0.13	1.83
FR Consumer Goods	24: 2 :1	1.07± 0.20	1.48
FR Total Production	28: 2 :1	1.07± 0.05	1.62

The TDNN trained on first-order differenced data sets produced lower RMSE than the TDNN* for all data sets, with improvement on USBC retail (64%), hardware (66%),

clothing (66%), furniture (24%), bookstore (46%), FR durable goods (51%), fuels (10%), consumer goods (28%), and total production (34%).

We found that the number of input delays in optimum TDNN architectures shown in Table III is highly correlated with the cycle information obtained from Fourier Analysis for each data set. We recently reported an algorithm to configure optimum TDNN architectures for analyzing cyclic series (Taskaya-Temizel, Casey, & Ahmad, 2005), finding that the number of input delays should be selected by taking into consideration the longest cycle information and the number of input weights in the network. On the five USBC data sets having size of 120, we found that there are no significant longer cycles than 12. If we assume a relaxation of ± 2 as per Zhang and Qi (2005), we can approximate the best performed TDNN input layer design in Table III. For longer series such as FR total production exhibiting several cycles such as 24, 37, 42 and 63 months, we undertook a similar experiment with configurations varying between $2i : 2j : 1$, where $1 \leq i, j \leq 33$. We observed that the network gives its best performance on $43 : 2 : 1$, which is close to 42 periods obtained from Fourier Analysis, but the performance degrades in larger input sizes, such as in 63. In addition, the number of hidden layer nodes should be kept small as generalization performance reduces for networks with larger hidden layers. This result agrees with the application of a TDNN to S&P financial time series (Sitte & Sitte, 2000). However, in these experiments the conclusion was that the input layer does not play a significant role in neural network design, perhaps due to the selected series following a random walk, in contrast to our results. These results also disagree with the application of a TDNN to the exchange rate data between British pound and US dollar (Zhang & Berardi, 2001). Although the time serial data exhibits a random walk, they conclude that forecasting ability of neural networks are not sensitive to the number of hidden nodes but sensitive to the number of input nodes.

In order to investigate the effect of the input and hidden layers on the overall performance of TDNNs, we calculated the mean RMSE of 30 randomly initialized TDNNs for each configuration. Fig. 1 illustrates the testing set performance of USBC bookstore time series. The x-axis and y-axis show the hidden and input layer sizes, respectively. The bar on the right side of the figure shows the correspondence between RMSE and shading, with dark depicting lower errors. It is apparent that the RMSE is significantly large when the input layer size is less than 12 (corresponds to a year) and that the best fit results are obtained for a lower number of neurons in the hidden layer. The error surface of FR fuels (see Fig. 2) shows similar results. In addition, the performance in the input layer degrades after 14 in Fig. 1, however not as much as the performance variation between networks having input layer with size of 12 and less. The other notable result is that the neural networks give good estimates when the input layer size is close to that of extracted cycle information. In Fig. 2, the darkest regions are clumped around 12 and 30, which are two of the cycles found in the FR fuels data set.

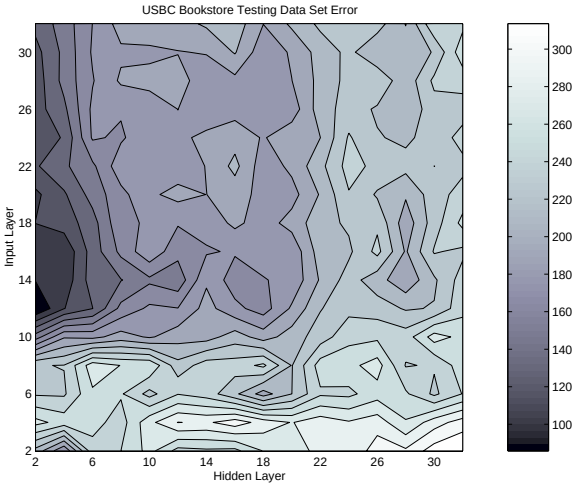

Fig. 1. USBC bookstore testing data set performance based on average RMSE. For each configuration, the mean RMSE is calculated over 30 trials. Mean RMSE is grouped into discrete bands to show the error landscape corresponding to the different layer sizes.

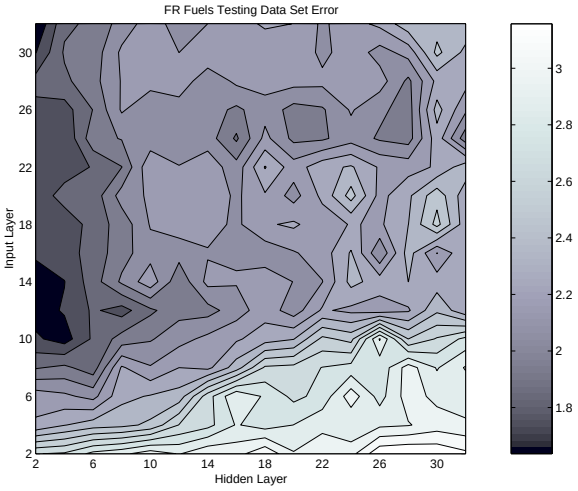

Fig. 2. FR fuels testing data set performance based on average RMSE. For each configuration, the mean RMSE is calculated over 30 trials. Mean RMSE is grouped into discrete bands to show the error landscape corresponding to the different layer sizes.

In addition, we found that there is no significant evidence that one should incorporate autocorrelation structures. Zhang and Qi (2005) included 10 various lag numbers of 1-4, 12-14, 24, 25 and 36 for the original and detrended data where seasonality exists. They attributed this to the observations being 12, 24 and 36 months apart, with high correlation, and hence it is necessary to include these lags in the input layer. However, the performances of the neural networks comprising 24 delays in the input layer did not yield the best results in our experiments.

C. Experiment 3: Performance comparison of linear autoregressive and neural networks

In this section, we compare the performance of linear models with neural networks. Linear AR models were constructed

using the AICC criteria as described in Section IV-B [2]. The performance of the validation set was used for model selection.

TABLE IV

AR MODEL PERFORMANCE: THE SECOND COLUMN SHOWS THE AR ORDER IDENTIFIED BY AICC. THE THIRD COLUMN PRESENTS THE RMSE OF THE AR MODEL. THE LAST COLUMN (*) IS THE RMSE OF ZHANG AND QI'S (2005) ARIMA MODEL

Data Set	AR Order	Test Error	ARIMA *
USBC Retail	11	551.84	1005.41
USBC Hardware	12	25.75	100.71
USBC Clothing	14	350.49	519.60
USBC Furniture	13	179.65	124.44
USBC Bookstore	12	111.17	98.17
FR Durable Goods	15	2.72	5.61
FR Fuels	13	1.53	1.62
FR Consumer Goods	13	0.97	3.96
FR Total Production	15	0.85	8.94

Our seven out of nine AR models performed considerably better than the ARIMA models constructed by Zhang and Qi (2005) (see Table IV). However the best fit results of the TDNN show a lower RMSE was obtained than AR fits of the USBC retail (6%), hardware (31%), clothing (28%), furniture (29%), bookstore (38%), FR durable goods (28%), fuels (45%), and consumer goods (13%) but not on total production (-1%). Our results show that a TDNN can outperform a linear model if the TDNN is configured appropriately.

D. Experiment 4: Are ARIMA neural network hybrids better than single models?

The hybrid architectures we tested were constructed following the procedure described in Section IV-C. We first detrended the time series and fitted linear AR to the detrended data, using the AR model orders shown in Table IV. Then we modeled the residuals of AR using 256 different TDNN architectures. Finally, for each data set, we selected the TDNN configurations that produced the best mean performance over 30 trials (lowest RMSE) from the 256.

AR hybrids performed better than single AR models on six out of nine data sets (compare Table IV and V). However, we observed a degeneration in performance in the USBC retail and clothing data sets. In the hybrid model, while the linear component is estimated by the AR model, the residual error (that is the error between the AR estimate and the original data) is estimated by a neural network. However, this residual error exhibits randomness, lacking the properties for a successful estimate by neural networks. The residual error and the residual error predicted by the neural network are shown in Fig. 3. This plot indicates that the error between the predicted residuals and the original residuals is greater than the error between zero (representing the case of no prediction, or just AR) and the residuals. This means that the neural network adversely affects the performance of the AR estimate, resulting in an overall poorer performance.

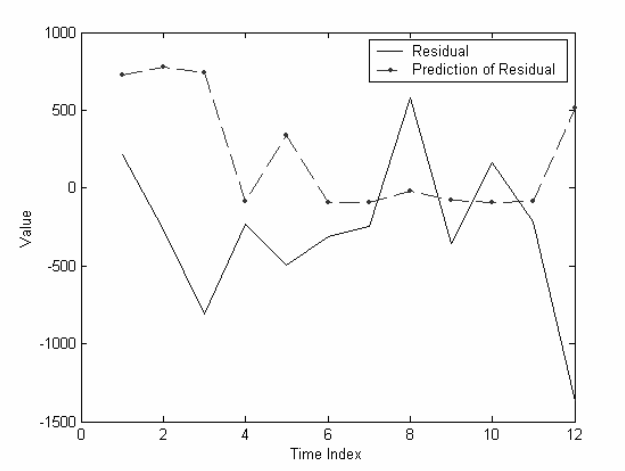

Fig. 3. The residuals of the AR process and the neural network prediction on the AR residuals. Although there is some correlation between the predictions, for higher time indices, the prediction differs significantly to the actual residual.

In Table V, we compared our hybrid model performances with Zhang and Qi's results. In their model construction, they used the X-11 method (current X-12-ARIMA) developed by the Bureau of the Census, which includes several seasonal adjustment methods. On average, we outperformed on three out of nine data sets. Recall that all our comparisons in the tables are based on mean and standard deviation of RMSE obtained over 30 trials, whilst Zhang and Qi's (2005) results are based on the best fit. Comparing our best fit results, we outperformed on six out of nine data sets: on the USBC retail (46%), hardware (63%), clothing (20%), furniture (11%), bookstore (22%), FR durable goods (46%), but not on fuels (-3%), consumer goods (-23%) and total production (-1%).

TABLE V

HYBRID ARCHITECTURE PERFORMANCE: THE SECOND COLUMN SHOWS THE BEST TDNN MODEL ORDER IDENTIFIED FOR THE RESIDUALS OF LINEAR AR. THE THIRD COLUMN IS THE MEAN RMSE OF THE HYBRID MODEL AND THE FOURTH COLUMN (*) IS THE BEST-FIT HYBRID RESULTS OF ZHANG & QI (2005)

Data Sets	Model	AR+TDNN	ARIMA+ TDNN*
USBC Retail	28: 2 :1	726.56±81.16	975.55
USBC Hardware	2:10:1	25.39± 3.80	49.17
USBC Clothing	2: 4 :1	381.76±10.70	315.43
USBC Furniture	22:10:1	173.03±19.55	99.45
USBC Bookstore	2: 6 :1	99.33± 6.99	88.74
FR Durable Goods	2: 2 :1	2.71± 0.05	3.63
FR Fuels	18: 4 :1	1.52± 0.12	0.81
FR Consumer Goods	4: 2 :1	0.98± 0.03	0.68
FR Total Production	2: 2 :1	0.83± 0.02	0.85

Another interesting result is that the optimum configurations of five out of nine of the TDNNs in the hybrid models have

[2]The Matlab programs to build autoregressive models and neural networks, as well as the autoregressive coefficients of the models, can be obtained from http://www.computing.surrey.ac.uk/personal/st/T.Taskaya/

similar input layer sizes. The tapped delays reveal that the AR models successfully removed the cyclic components from the differenced series, whilst the residuals appeared to follow a non-linear random walk model. For the three sets USBC retail, furniture and FR fuels, the TDNN configurations show that AR models could not successfully remove the long time cycles from the time series.

Fig. 4 shows the percentage performance improvement for the mean and best fit of the TDNN, best fit of the AR neural network hybrid and AR single models as compared to the mean of hybrid architecture. For four out of the nine data sets, the mean hybrid outperforms the single model. However, for five of the data sets, either the linear AR or TDNN model outperforms the hybrid. Of these improved single models, three significantly outperform the hybrid. These improvements appear to be related to model configuration, where selection for generalization performance allows for better results.

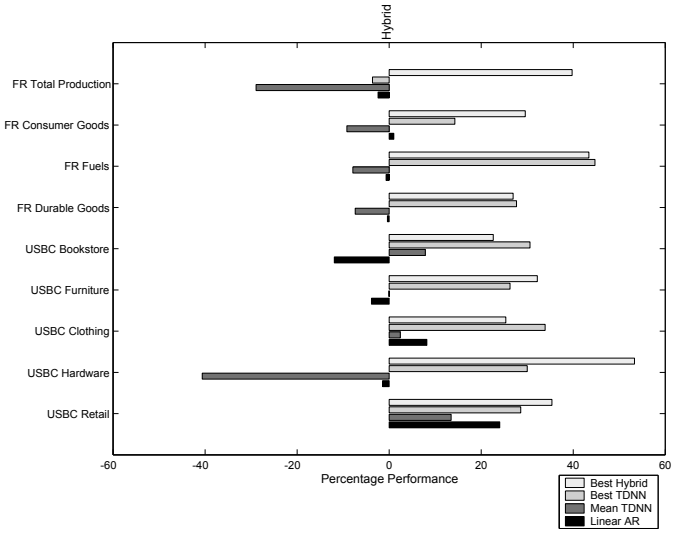

Fig. 4. Percentage performance improvement for the mean and best fit of the TDNN, best fit of the AR neural network hybrid and AR single models as compared to the mean for the hybrid architecture.

VI. Discussion

In this paper, we have attempted to understand if hybrid models really are better than single models. Our findings can be summarized as follows:

A) How important is preprocessing for neural networks? How does detrending affect the performance?

If a given time series exhibits trend, one should employ detrending. The selection of the detrending process is also vital for training with neural networks. We found that differencing generally gives superior results than that of trend fitting.

B) Are neural networks able to model seasonality? If they are, how can we construct optimal architectures?

Neural networks are able to model seasonality if the neural network architecture is properly configured. It appears that neural networks give better forecasts when the input layer size is equal to the longest cycle information obtained from Fourier Analysis (Taskaya-Temizel et al., 2005) and the hidden layer size is small, relating to the generalization capabilities of the network.

C) Compared to linear autoregressive models, how successful are neural networks?

When the mean RMSE of the TDNNs are compared to linear AR processes, they outperform in two out of the nine data sets. When the best fit results are compared, the TDNNs outperform the AR processes in eight out of nine data sets. However, to obtain these better results requires effort in configuring the network appropriately.

D) Are ARIMA neural network hybrids better than single models?

For five of the nine data sets, the linear AR and TDNN models outperform the ARIMA neural network hybrids, albeit with similar levels of performance for two of these data sets. This demonstrates that, despite the popularity of hybrid models, which rely upon the success of their components, single models themselves *can* be sufficient. Perhaps the danger in using ARIMA neural network hybrids is that there is an assumption that the relationship between the linear and non-linear components is additive and this may degrade performance if the relationship is different (for example multiplicative). In addition, one may not guarantee that the residuals of the linear component may comprise valid non-linear patterns.

These results show that hybrids are not always better, and hence that the model selection process still remains an important step despite the popularity of hybrid models. We have focused on a limited subset of hybrid models, and therefore further work is required to assess the generated performance of hybrid models in comparison to single models. Following on from these results, there are still some questions to be answered. For example, we also plan to work on model selection procedures for TDNN architectures. In our earlier work, we found that there is a strong relationship between the cycle information obtained from Fourier Analysis and the number of weights in the neural networks. We will further investigate the impact that this has on generalization capability.

Acknowledgments

We are grateful to Fingrid project [RES-149-25-0028], which provided a Grid environment comprising 24 machines to run our simulations using Condor. We would like to thank to Dr. Anthony Browne, and Alptekin Temizel for their proofreading and comments.

VII. References

Bishop, C. (1994). Mixture density networks. *Neural Computing Research Group Report: NCRG/94/004*, 1–25.

Box, G.-E.-P., & Cox, D.-R. (1996). An analysis of transformations. *JRSS B*, *26*, 211–246.

Brown, G., Wyatt, J., Harris, R., & Yao, X. (2005). Diversity creation methods: a survey and categorisation. *Information Fusion*, *6*(1), 5–20.

Chatfield, C. (2004). *The analysis of time series*. Texts in Statistical Science. USA: Chapman & Hall, sixth edition.

Clemen, R. (1989). Combining forecasts:a review and annotated bibliography. *International Journal of Forecasting*, *5*, 559–583.

Clouse, D., Giles, C., & Horne, G. (1997). Time delay-neural networks: Representation and induction of finite-state machines. *IEEE Transactions on Neural Networks*, *8*(5), 1065–1070.

Cottrell, M., Girard, B., Girard, Y., Mangeas, M., & Muller, C. (1995). Neural modeling for time series:a statistical stepwise method for weight elimination. *IEEE Transactions on Neural Networks*, *6*(6), 1355–1364.

Elman, J. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Hansen, J., & Nelson, R. (2003). Time-series analysis with neural networks and ARIMA-neural network hybrids. *Journal of Experimental and Theoretical Artificial Intelligence*, *15*(3), 315–330.

Harvey, A. (1997). Trends, cycles and autoregressions. *The Economic Journal*, *107*(1), 192–201.

Haykin, S. (1999). *Neural networks: A comprehensive foundation*. Prentice Hall, 2 edition.

Hibbert, H., Pedreira, C., & Souza, R. (2000). Combining neural networks and ARIMA models for hourly temperature forecast. *Proceedings of International Conference on Neural Networks (IJCNN 2000)* (pp. 414–419). Como,Italy.

Hibon, M., & Evgeniou, T. (2005). To combine or not to combine: Selecting among forecasts and their combinations. *International Journal of Forecasting*, *21*, 15–24.

Hurvich, C., Simonoff, J., & Tsai, C.-L. (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society: Series B*, *60*(2), 271–293.

Jacobs, R., Jordan, M., Nowlan, S., & Hinton, G. (1991). Adaptive mixture of local experts. *Neural Computation*, *3*, 79–87.

Kantz, H., & Schreiber, T. (1999). *Non-linear time series analysis*. Cambridge, UK: Cambridge University Press.

Nelson, M., Hill, T., Remus, W., & O'Connor, M. (1999). Time series forecasting using neural networks: Should the data be deseasonalized first? *Journal of Forecasting*, *18*, 359–367.

Rao, T.-S., & Sabr, M.-M. (1984). An introduction to bispectral analysis and bilinear time series models. *Lecture Notes in Statistics*, *24*.

Sharkey, A. (2002). Types of multinet system. *Proceedings of the Third International Workshop on Multiple Classifier Systems*, Vol. 2364 of *Lecture Notes in Computer Science* (pp. 108–117). London, UK: Springer-Verlag.

Sitte, R., & Sitte, J. (2000). Analysis of the predictive ability of time delay neural networks applied to the S&P 500 time series. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, *30*, 568–572.

Taskaya-Temizel, T., & Ahmad, K. (2005). Are ARIMA neural network hybrids better than single models? *Proceedings of International Joint Conference on Neural Networks (IJCNN 2005)*. July 31 - August 4, 2005, Montréal, Canada.

Taskaya-Temizel, T., Casey, M.-C., & Ahmad, K. (2005). Pre-processing inputs for optimally-configured time-delay neural networks. *IEE Electronic Letters*, *41*, 198–200.

Terui, N., & van Dijk, H. (2002). Combined forecasts from linear and nonlinear time series models. *International Journal of Forecasting*, *18*, 421–438.

Tong, H. (1990). *Non-linear time series: a dynamical system approach*. Oxford University Press.

Tseng, F.-M., Yu, H.-C., & Tzeng, G.-H. (2002). Combining neural network model with seasonal time series ARIMA model. *Technological Forecasting and Social Change*, *69*, 71–87.

Virili, F., & Freisleben, B. (2000). Nonstationarity and data preprocessing for neural network predictions of an economic time series. *Proceedings of International Joint Conference on Neural Networks (IJCNN 2000)* (pp. 5129–5136). Como, Italy.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K.-J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *37*(3), 328–339.

Weigend, A.-S., Mangeas, M., & Srivastava, A.-N. (1995). Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. *International Journal of Neural Systems*, *6*, 373–399.

Wichard, J., & Ogorzalek, M. (2004). Time series prediction with ensemble models. *Proceedings of International Joint Conference on Neural Networks (IJCNN 2004)* (pp. 1625–1629). Budapest.

Wong, C., & Li, W. (2000). On a mixture autoregressive model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *62*, 91–115.

Zhang, G.-P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*, 159–175.

Zhang, G.-P., & Berardi, V. (2001). Time series forecasting with neural network ensembles: an application for exchange rate prediction. *Journal of the Operational Research Society*, *52*, 652–664.

Zhang, G.-P., & Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, *160*(2), 501–514.